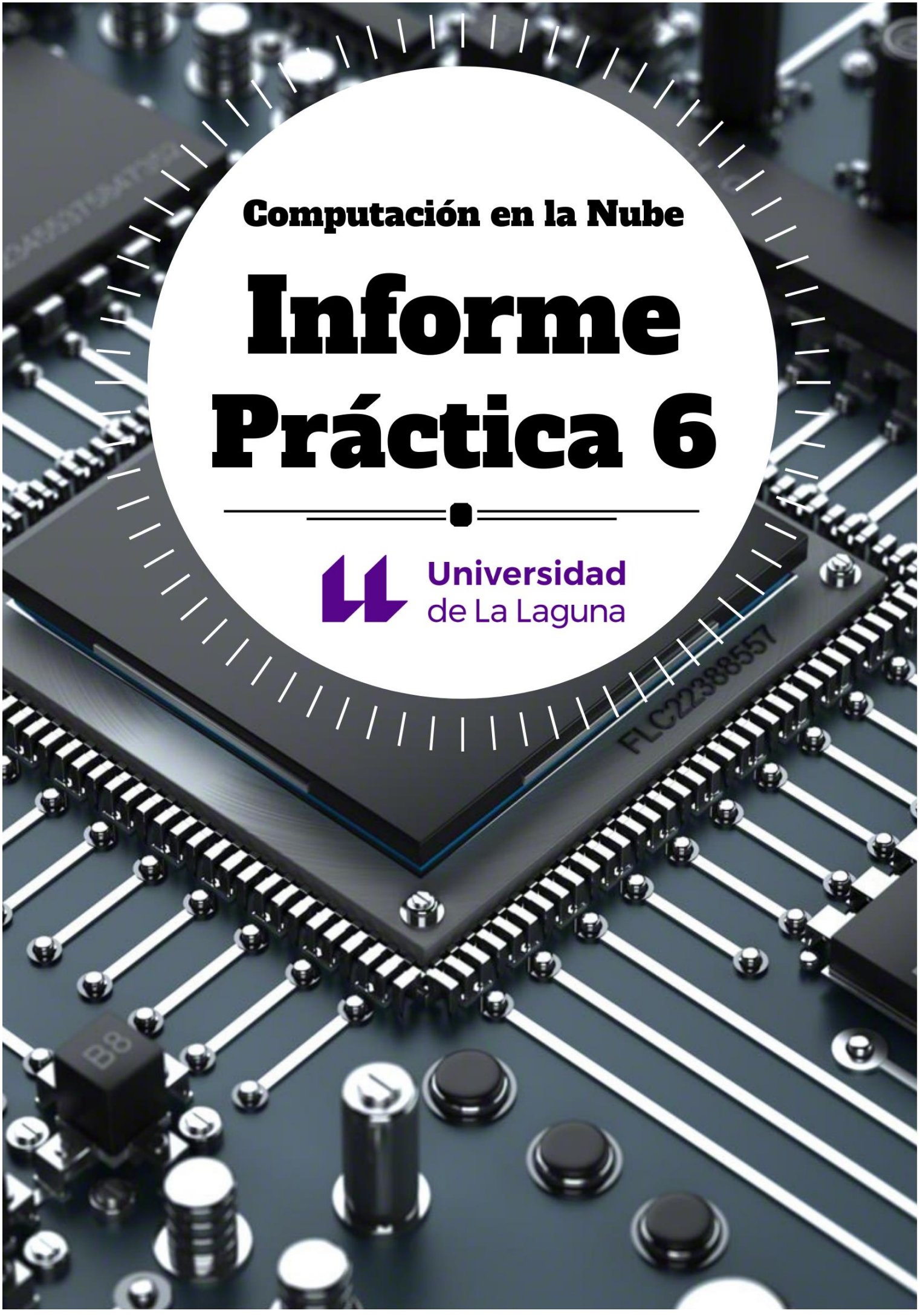


Computación en la Nube

Informe Práctica 6

Universidad de La Laguna

INDICE

1. Framework de visualización.

1.1 Propuestas realizadas por el profesor.

1.1.1 Autentificación.

Para este apartado se han realizado una nueva funcionalidad en la api que permite autentificar a un usuario (registrado con anterioridad). Para ello, se ha hecho uso de mongo y "jsonwebtoken". Esta funcionalidad la podremos ver a continuación:

Backend

```
async function authenticate({ email, password }) {
    const user = await User.findOne({ email })

    if (user && bcrypt.compareSync(password, user.password)) {
        const token = jwt.createToken(user)
        return {
            token
        };
    }

}
```

Frontend

De igual forma, y apoyado por las funciones anteriores, se ha creado una seguridad adicional para la funcionalidad de añadir un nuevo algoritmo. Para este caso, es necesario recibir el usuario y contraseña, posteriormente verificarlos para poder subir dicho algoritmo. Esto lo podremos ver a continuación:

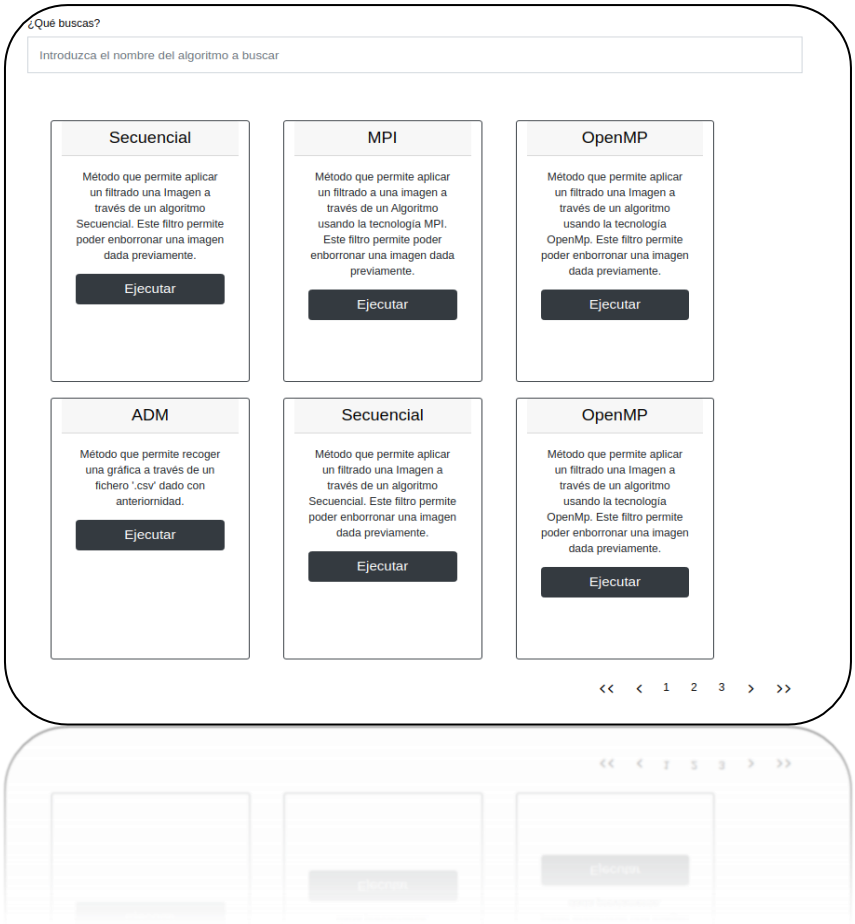
```
// Permite Autentificarte
app.post('/api/authenticate/:email/:passwd', (req, res) => {
  axios.post('/users/authenticate', {
    "email": req.params.email,
    "password": req.params.passwd
  })
  .then((res2) => {
    var json = '{"Token":"' + res2.data.token + '"}';
    fs.readFile("./app/src/Servidor/Token.json", 'utf-8', (err2, data) => {
      var obj = JSON.parse(data);
      obj['Tokens'].push(JSON.parse(json));
      jsonStr = JSON.stringify(obj);
      fs.writeFileSync('./app/src/Servidor/Token.json', jsonStr, { mode: 0o755 });
    });
    res.send(res2.data.token);
  })
  .catch((error) => {
    res.send(false);
  })
});

// Permite Crear Métodos
app.post('/api/Upload/Method/:token', uploadMethod.array('file', 2), (req, res) => {
  fs.readFile("./app/src/Servidor/Token.json", 'utf-8', (err, data) => {
    var obj = JSON.parse(data);
    var position = obj["Tokens"].findIndex(element => element.Token === req.params.token);

    if(position == -1){
      res.send(false);
    }
    else{
      fs.readFile("./app/src/Servidor/EstructuraMetodos/" + req.files[0].filename, 'utf-8', (err, data) => {
        fs.readFile("./app/src/Servidor/Metodos.json", 'utf-8', (err2, data2) => {
          var obj = JSON.parse(data2);
          obj['Methods'].push(JSON.parse(data));
          jsonStr = JSON.stringify(obj);
          fs.writeFileSync('./app/src/Servidor/Metodos.json', jsonStr, { mode: 0o755 });
          var NewMethod = JSON.parse(data);
          executeUnzip('./app/src/Servidor/EstructuraMetodos/'+ req.files[1].filename, './app/src/Servidor/Metodos/' + NewMethod["Name"]);
          executeMake(NewMethod["Name"]);
        });
      });
      res.send(true);
    }
  });
});
```

1.1.2 Filtrado algoritmos.

Tras realizar la corrección la semana pasada, el profesor propuso una pequeña mejora. Esta, consiste en la posibilidad de realizar un filtrado de los algoritmos para facilitar su búsqueda. Para realizar esto, se ha optado por incorporar un nuevo input que nos permita filtrar los algoritmos a tiempo real (sin pulsar ningún botón). De igual forma, se ha añadido una paginación permitiendo ver solo 6 algoritmos a la vez. Esto lo podremos ver a continuación:



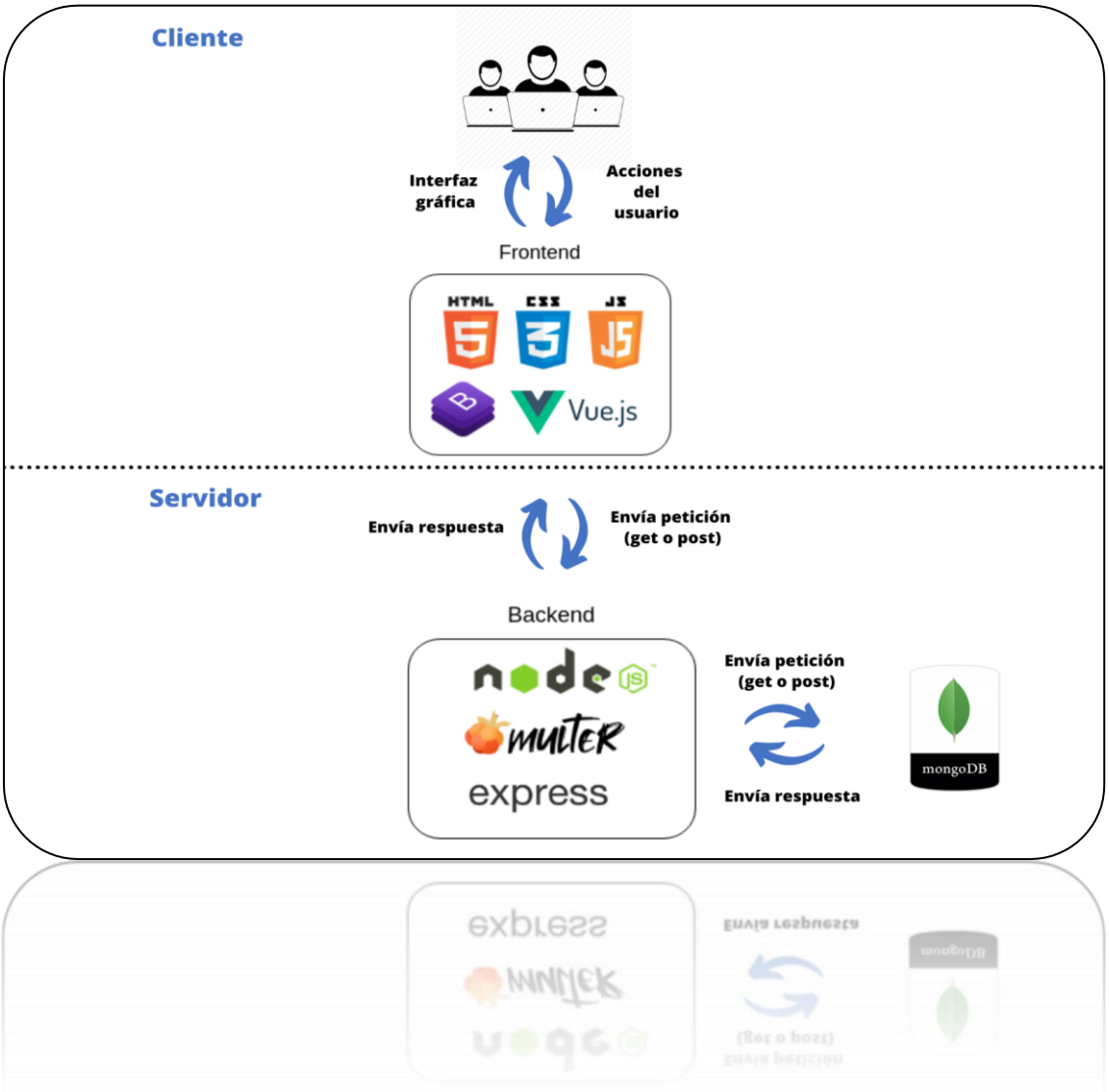
1.2 Propuestas realizadas por el alumno.

1.2.1 Cambio Tecnología usada.

Tras ver un límite en el despliegue de una aplicación sin un stack de desarrollo, se ha procedido a cambiar la filosofía. Tras analizar detenidamente la cantidad de software existente, se ha escogido el que nos proporciona MEVN, esta elección se debe a que nos proporciona muchas ventajas, entre las que cabe destacar: presentar previa experiencia en esta y su sistema de componentes. Este stack está compuesto de:

- **Mongodb**: base de datos de tipo no SQL que nos posibilita el manejo de grandes volúmenes de datos gracias a la tecnología de JSON.
- **Express**: framework que nos permite facilitar la creación de API REST, puesto que contiene muchas de las funcionalidades de estas y otras aportaciones que nos benefician.
- **Vue.js**: framework de desarrollo de la una página web que implementa la idea de usar Javascript en prácticamente todo el proyecto.
- **Node.js**: entorno de programación del lado del servidor que nos proporciona la posibilidad de mostrar una página web en el navegador.

A continuación, podremos ver un gráfico representando la nueva tecnología usada:



1.2.2 Nuevas acciones del usuario.

Además de todas las nuevas características de esta entrega se han añadido diversas funcionalidades nuevas para el usuario. Estas las podremos ver a continuación:

- **Página de Inicio:** se ha creado una nueva página de inicio más estética que explica que es “CodeCloud”. De igual forma, podremos ver un carrusel con imágenes sobre noticias de interés para la comunidad y otro en el que podremos ver 9 algoritmos y nos permite acceder a ver el resto. Esto lo podremos ver a continuación:

- **Registrar:** funcionalidad que nos permite crear a un nuevo usuario a partir de los siguientes datos:

```
const userSchema = new Schema({
    // username: { type: String, unique: true, required: true },
    password: { type: String, required: true },
    name: { type: String, required: true },
    surname: { type: String, required: true },
    email: { type: String, unique: true, required: true },
    paragraph: { type: String, required: true },
    telephone: { type: String, required: true },
    birthdate: String,
    genre: { type: String },
    createdDate: { type: Date, default: Date.now }
});
```

Esta funcionalidad la podremos ver a continuación:

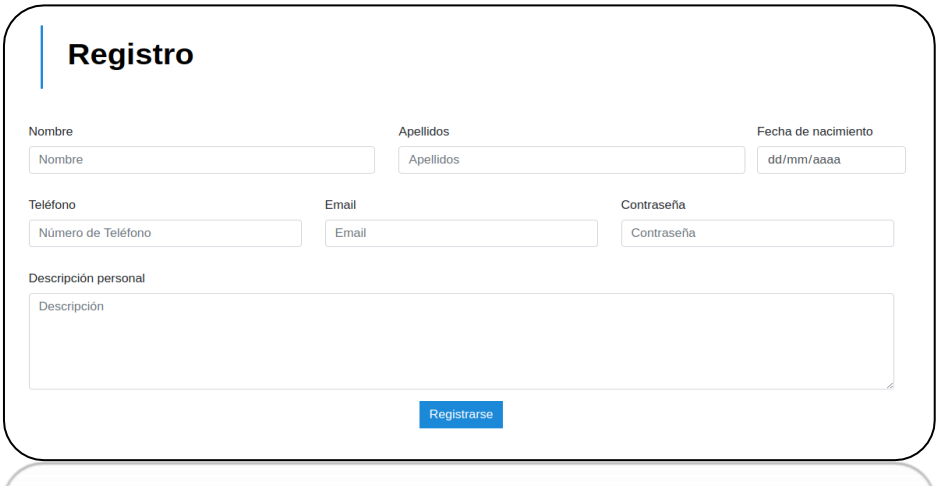
Backend

```
async function create(userParam, res) {

    if (await User.findOne({ email: userParam.email })) {
        throw Error(`El email ${userParam.email} ya está registrado`)
    }
    else {
        const user = new User(userParam)

        if (userParam.password) {
            user.password = bcrypt.hashSync(userParam.password)
        }
        await user.save()
    }
}
```

Frontend



- **Actualizar:** funcionalidad que nos permite actualizar los datos de un usuario específico (nombre,..). Esta funcionalidad la podremos ver a continuación:

Backend

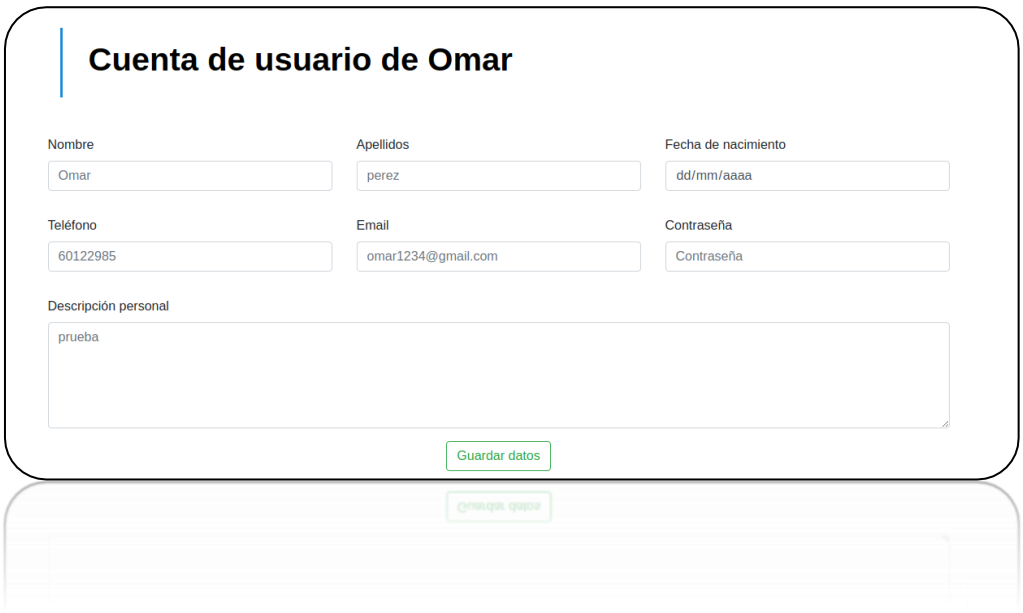
```
async function update(id, userParam) {
    const user = await User.findById(id)

    if (!user) throw 'User not found'
    if (user.email !== userParam.email && await User.findOne({ email: userParam.email })) {
        throw Error(`email ${userParam.email} is already taken`)
    }

    if (userParam.password) {
        user.password = bcrypt.hashSync(userParam.password, 10)
    }
    if(userParam.name) {
        user.name = userParam.name
    }
    if(userParam.surname) {
        user.surname = userParam.surname
    }
    if(userParam.email) {
        user.email = userParam.email
    }
    if(userParam.paragraph) {
        user.paragraph = userParam.paragraph
    }
    if(userParam.image) {
        user.image = userParam.image
    }
    if(userParam.telephone) {
        user.telephone = userParam.telephone
    }
    if(userParam.birthdate) {
        user.birthdate = userParam.birthdate
    }
    if(userParam.genre) {
        user.genre = userParam.genre
    }

    await user.save()
}
```

Frontend



- **Eliminar:** funcionalidad que nos permite eliminar a un usuario específico. Esta la podremos ver a continuación:

Backend

```
async function _delete(id) {
    await User.findByIdAndRemove(id)
}
```

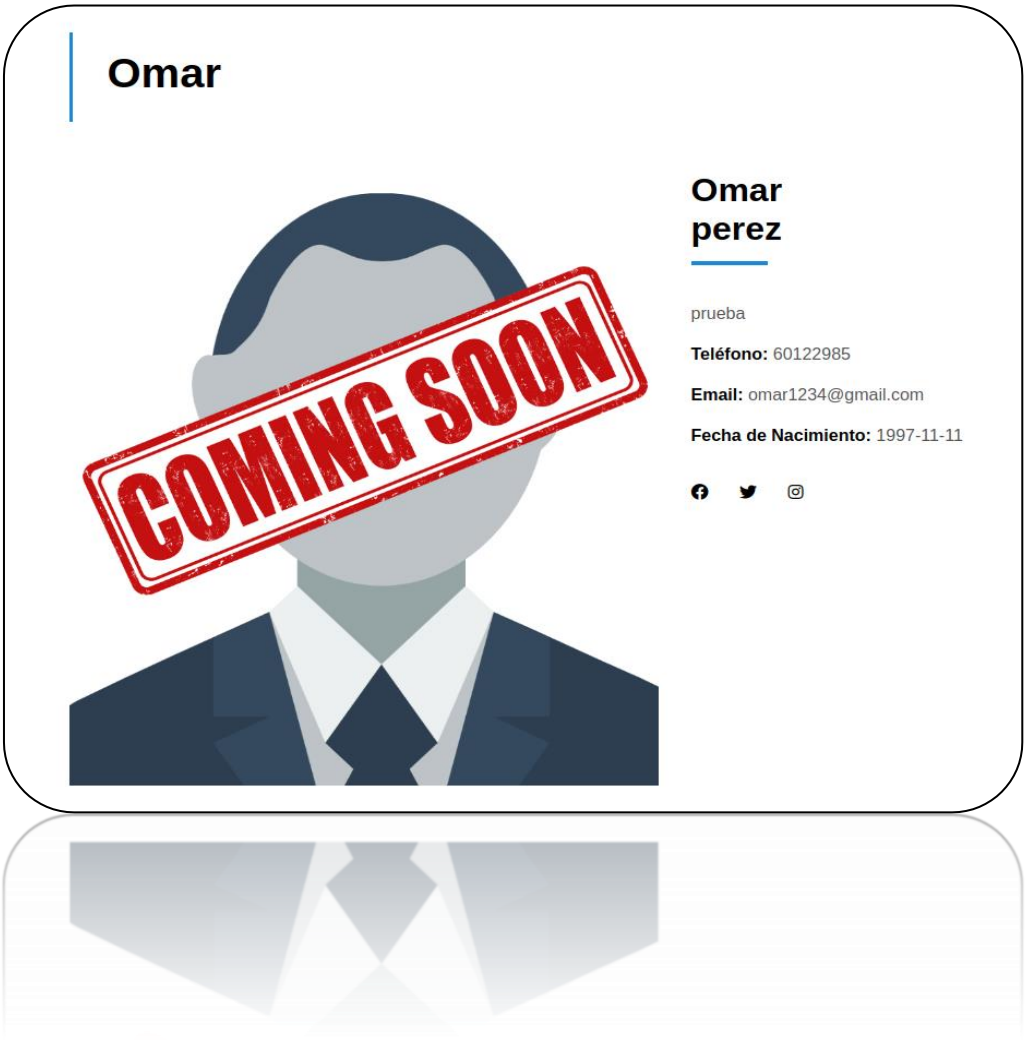
Frontend

- **Actual usuario:** funcionalidad que (a partir de un id) nos permite recoger los datos de un usuario específico (sin la contraseña por seguridad).

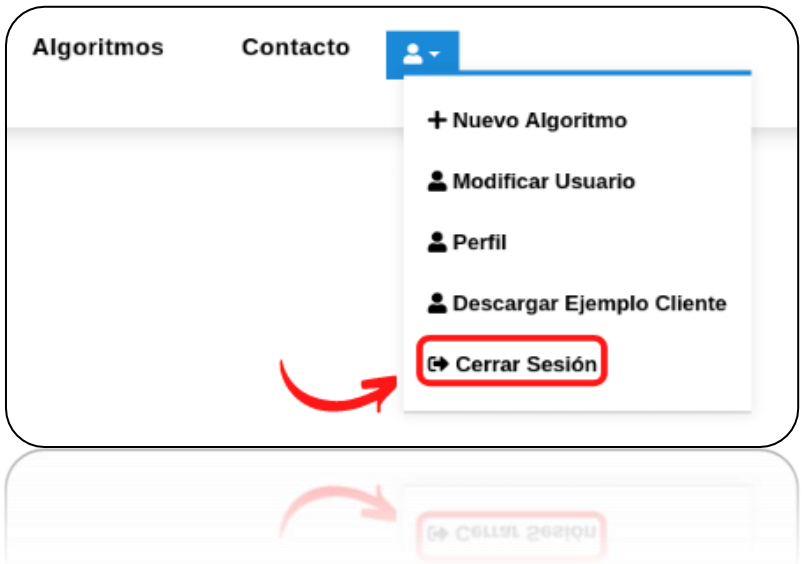
Backend

```
async function getById(id) {
    return await User.findById(id).select('-password')
}
```

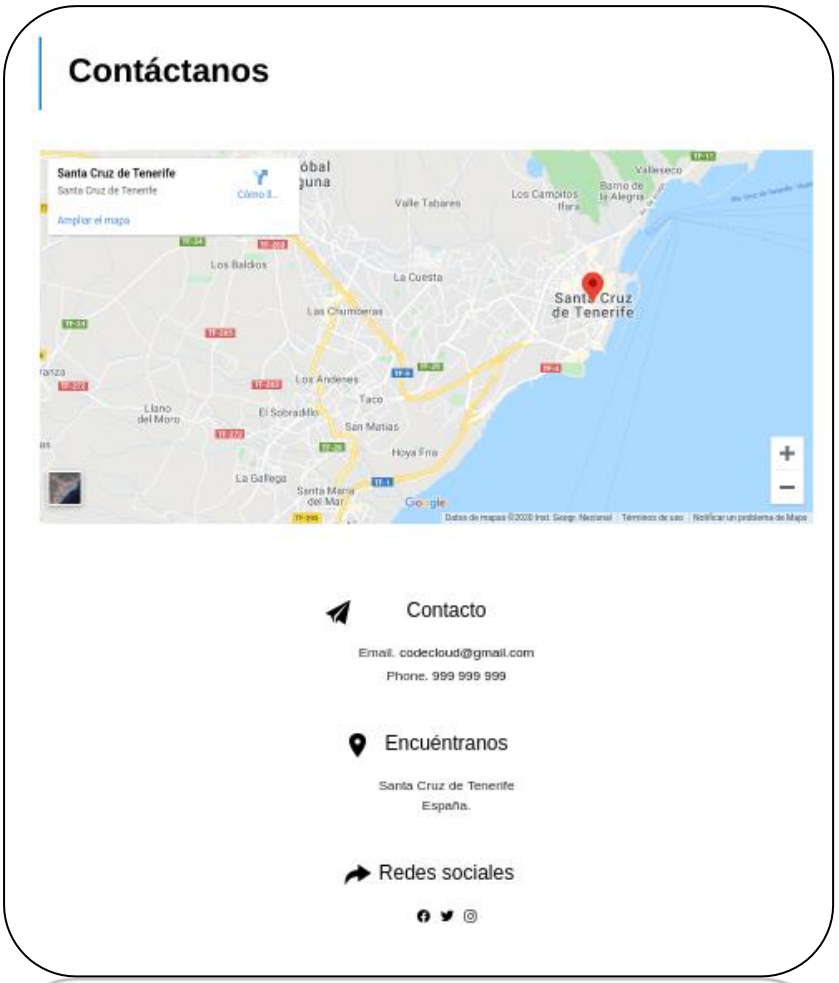
Frontend



- **Cerrar sesión:** funcionalidad que nos permitirá cerrar la sesión (borrar el token de la sesión). Esto lo podremos ver a continuación:



- **Contacto:** página que simula una página de contacto a una empresa. Esta la podremos ver a continuación:

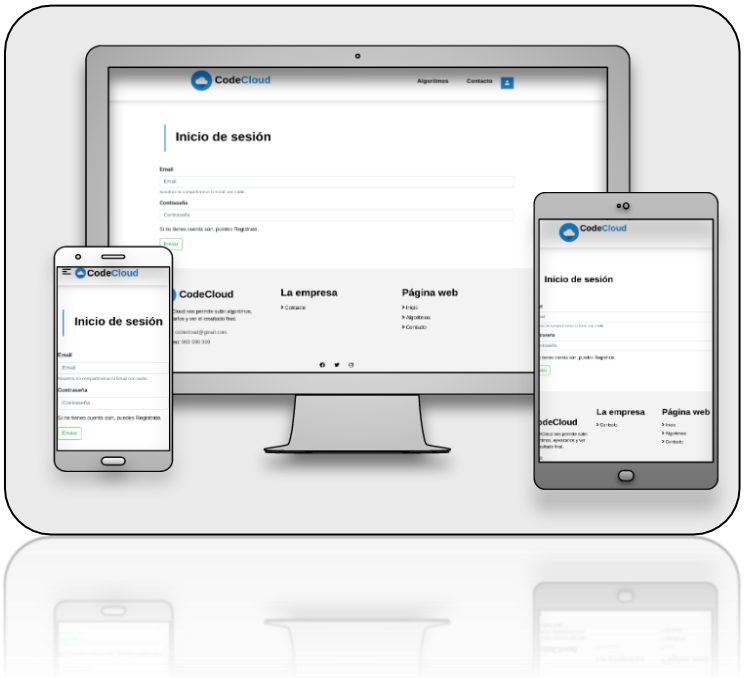


1.2.3 Diseño Responsive.

A medida que se han ido diseñando cada página vista con anterioridad se ha pensado en el Diseño Responsive. A continuación, podremos ver algunos ejemplos:

Página Inicio

Inicio sesión



2. Utilidad con vistas a futuro.

Esta aplicación dispone de diversas opciones para poder desplegarla en el futuro. En primera instancia, se han pensado las siguientes características:

- Capacidad de disponer un perfil donde veremos diversos datos sobre la persona. Así como, sus algoritmos más usados, mejores votados y diversas funcionales de la misma ideología.
- Capacidad de poder modificar nuestro perfil con estilos más llamativos. Para ello, se ha pensado en que el usuario pueda subir su código html y css, para así poder modificar la estética de su perfil.

Junto con lo anterior, la idea principa es aumentar el atractivo de tu currículum online en diversas plataformas existentes (infojobs,..) y, así diversas empresas tengan más información sobre la persona que se quiere contratar.

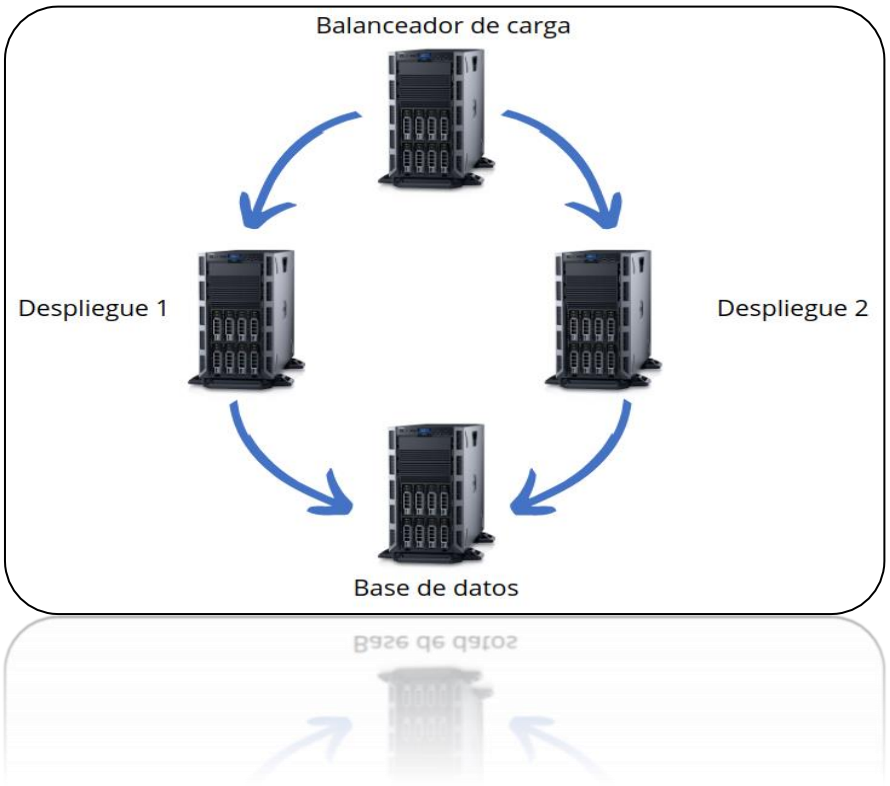
Como añadido a lo anteriormente comentado, se ha pensado crear una integración con un repositorio de los diversos sistemas de control de versiones (github,..). Y así, con cada cambio realizado en estos se actualiza automáticamente en nuestra página web.

Junto con lo comentado, se ha pensado poder seguir avanzando y no solo quedarnos en el cómputo de algoritmos online. Sino también en la posibilidad de crear los algoritmos de forma online (IDE online) y poder (a través de html y css) darle una parte gráfica a estos. De igual forma, se podría no solo centrarse en los algoritmos, sino que, también en la posibilidad de crear páginas webs estáticas (sólo html y css) que se pudieran ver (para añadir mayor información a tu currículum) e incluso vender al público.

3. Análisis de costes despliegue.

3.1 Estructura de despliegue.

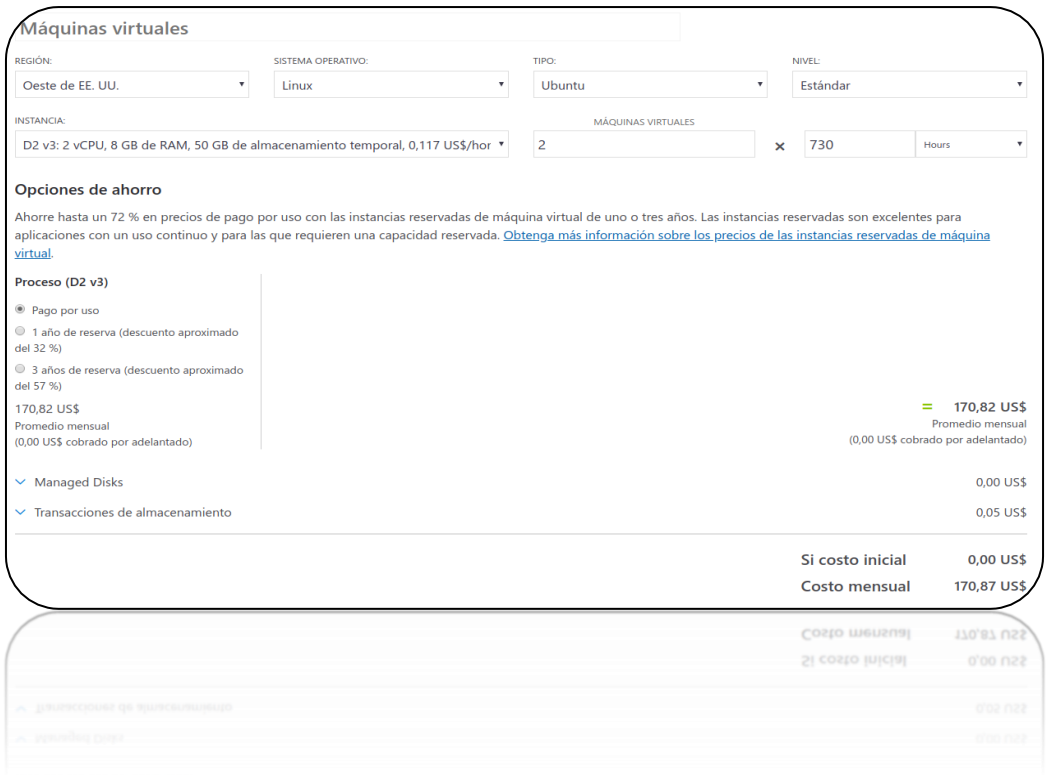
Para este apartado, se ha escogido la empresa Microsoft. Por tanto, se ha realizado un análisis a la tecnología denominada Microsoft Azure. En primera instancia, se ha pensado una estructura inicial que se irá aumentando a medida que se vaya necesitando (cada vez que sea más rentable nuestra aplicación). Por ello, se ha propuesto el siguiente esquema:



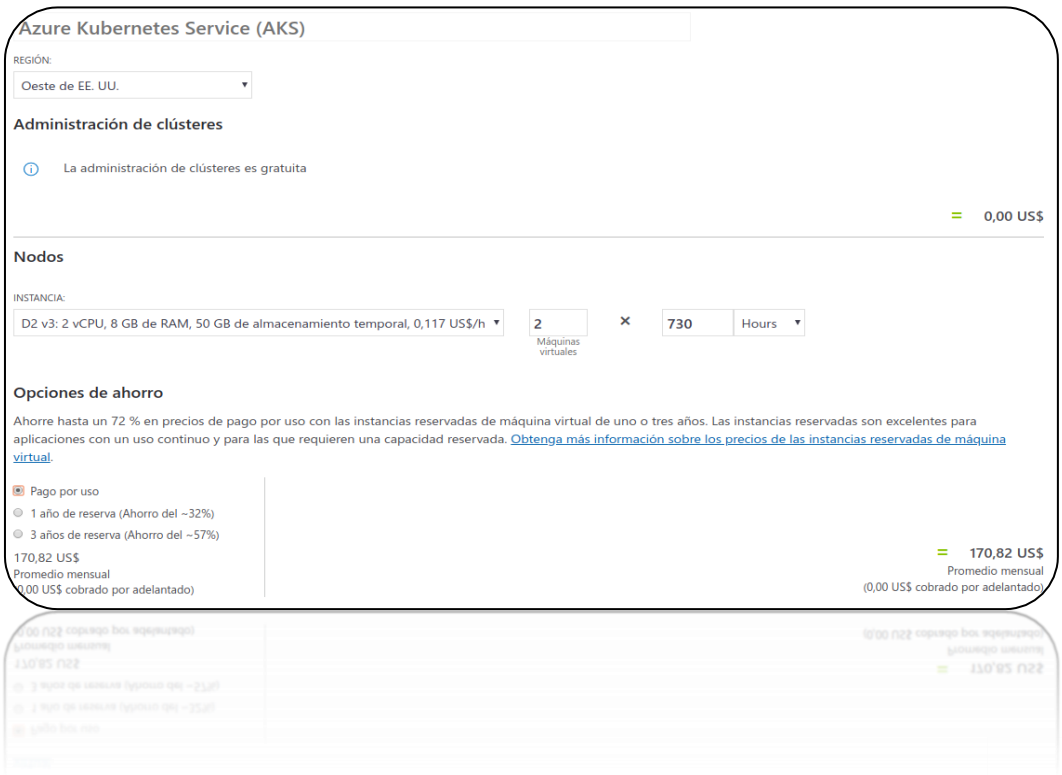
3.2 Costos de despliegue

Como vimos en el apartado anterior, dispondremos de un balanceador de carga que nos permitirá repartir la carga entre dos despliegues diferentes y capacidad de tolerancia a fallos, es decir, si un despliegue falla tendremos el otro para soportar dicha carga hasta que se solucione. Para permitir desplegar la estructura anterior, existen varias opciones, algunas de estas las podremos ver a continuación.

- Alquilar 2 máquinas y crear contenedores (de forma manual) en su interior. Estas máquinas tendrían 2vcpu, 8gb de RAM y 50gb de almacenamiento. Y su precio sería de 0,117 $/hora por cada máquina. Esto nos quiere decir que su precio mensual sería de 170,87$ al mes (si se tiene encendido todo el día). Sin embargo, si se contratan 1 año asegurado este precio bajaría a 115,84 $ al mes y, si se contratan 3 años el precio final sería de 74,11 $ mensuales. Esto lo podremos ver a continuación:



- Contratar el servicio Azure Kubernetes junto con 2 máquinas virtuales. Estas máquinas tendrían 2vcpu, 8gb de RAM y 50gb de almacenamiento. Y su precio sería de 0,117 $/hora. Esto nos quiere decir que su precio mensual sería de 170,82$ al mes (si se tiene encendido todo el día). Sin embargo, si se contratan 1 año asegurado este precio bajaría a 115,84 $ al mes y, si se contratan 3 años el precio final sería de 74,11 $ mensuales. Esto lo podremos ver a continuación:



3.3 Pensamiento a largo plazo.

Tras hacer el análisis de los precios (apartado anterior), es mejor opción contratar un servicio con Azure Kubernetes (con las dos máquinas virtuales). De igual forma, esta tecnología nos permite crear más máquinas a medida que sube la carga (hasta un límite). Esta característica nos permitirá no solo poder aguantar cargas altas en un periodo concreto de horas, sino que, también nos asegura la capacidad de crecer a corto, medio y largo plazo. A su vez, el costo de aumentar una máquina consta de 170,82$ (sin permanencia), 115,84$ (reservando un año) y 74,11 $ (reservando 3 años).